



日 本 国 特 許 庁  
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出 願 年 月 日

Date of Application:

2001年 1月18日

出 願 番 号

Application Number:

特願2001-009650

出 願 人

Applicant(s):

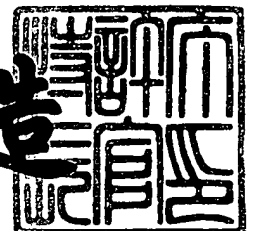
株式会社日立製作所

CERTIFIED COPY OF  
PRIORITY DOCUMENT

2001年11月16日

特許庁長官  
Commissioner,  
Japan Patent Office

及 川 耕 造



【書類名】 特許願

【整理番号】 K01000691A

【提出日】 平成13年 1月18日

【あて先】 特許庁長官殿

【国際特許分類】 H04L 12/46

【発明者】

    【住所又は居所】 神奈川県横浜市戸塚区戸塚町 5 0 3 0 番地 株式会社日立製作所 ソフトウェア事業部内

    【氏名】 高木 渉

【発明者】

    【住所又は居所】 神奈川県横浜市戸塚区戸塚町 5 0 3 0 番地 株式会社日立製作所 ソフトウェア事業部内

    【氏名】 横塚 大典

【特許出願人】

    【識別番号】 000005108

    【氏名又は名称】 株式会社日立製作所

【代理人】

    【識別番号】 100075096

    【弁理士】

    【氏名又は名称】 作田 康夫

【手数料の表示】

    【予納台帳番号】 013088

    【納付金額】 21,000円

【提出物件の目録】

    【物件名】 明細書 1

    【物件名】 図面 1

    【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 構造化文書をプログラム言語の構造体データへマッピングするシステム及び方法及びプログラム

【特許請求の範囲】

【請求項 1】

構造化文書のデータ構造からプログラム言語のデータ構造へデータを転記するプログラムであって、

前記プログラムは、構造化文書の構造を定義した情報と、プログラム言語の構造体の定義情報と、構造化文書の構造とプログラム言語の構造体との対応情報とを取得し、

前記取得した情報に基づいて、前記プログラム言語の構造体の状態を格納しておくフラグ構造体と前記フラグ構造体に対応するプログラム言語の構造体を作成し、

前記作成したフラグ構造体と前記フラグ構造体に対応するプログラム言語の構造体とを用いて、構造化文書のデータ構造からプログラム言語のデータ構造へデータを転記することを特徴とするデータ転記プログラム。

【請求項 2】

プログラム言語のデータ構造から構造化文書のデータ構造へデータを転記するプログラムであって、

前記プログラムは、構造化文書の構造を定義した情報と、プログラム言語の構造体の定義情報と、構造化文書の構造とプログラム言語の構造体との対応情報とを取得し、

前記取得した情報に基づいて、前記プログラム言語の構造体の状態を格納しておくフラグ構造体と前記フラグ構造体に対応するプログラム言語の構造体を作成し、

前記作成したフラグ構造体と前記フラグ構造体に対応するプログラム言語の構造体とを用いて、プログラム言語のデータ構造から構造化文書のデータ構造へデータを転記することを特徴とするデータ転記プログラム。

【請求項 3】

前記フラグ構造体とは、前記プログラム言語の構造体に含まれるデータの有無、データの出現回数、データの型、データの長さを有することを特徴とする請求項 3 記載のデータ転記プログラム。

【請求項 4】

企業間システムにおけるデータ構造を変換するプログラムであって、

前記企業間で共通に用いる構造化文書のデータ構造を定めておき、

前記プログラムは、前記企業間で用いる構造化文書のデータ構造を定義した情報と、企業内で用いるプログラム言語の構造体の定義情報と、前記企業間で用いる構造化文書のデータ構造と前記企業内で用いるプログラム言語の構造体との対応情報とを取得し、

前記取得した情報に基づいて、前記プログラム言語の構造体の状態を格納しておくフラグ構造体と前記フラグ構造体に対応するプログラム言語の構造体を作成し、

前記作成したフラグ構造体と前記フラグ構造体に対応するプログラム言語の構造体とを用いることを特徴とする、データ構造を変換するプログラム。

【請求項 5】

計算機で用いる構造化文書のデータ構造からプログラム言語のデータ構造へデータを転記する方法であって、

前記計算機は、構造化文書の構造を定義した情報と、プログラム言語の構造体の定義情報と、構造化文書の構造とプログラム言語の構造体との対応情報とを取得し、

前記取得した情報に基づいて、前記プログラム言語の構造体の状態を格納しておくフラグ構造体と前記フラグ構造体に対応するプログラム言語の構造体を作成し、

前記作成したフラグ構造体と前記フラグ構造体に対応するプログラム言語の構造体とを用いて、構造化文書のデータ構造からプログラム言語のデータ構造へデータを転記することを特徴とするデータ転記方法。

【発明の詳細な説明】

【 0 0 0 1 】

## 【発明の属する技術分野】

本発明は、プログラム言語で外部データを扱うシステム及び方法に関し、特にプログラム言語で構造化文書を扱うシステム及び方法およびプログラムに関する。

## 【0002】

## 【従来の技術】

XML (eXtensible Markup Language) に代表される構造化文書は、さまざまな用途に使われるが、特に、企業間のデータ交換形式として使われるとき、プログラム言語でデータを扱う必要がある。

## 【0003】

構造化文書自体は、文字列という以上に物理構造を持たないが、文字列内の文法規則によって構造を表現する。開始タグと終了タグの構文を用意し、これらのタグで文字列データを囲むことで構造化文書を構成する要素を表す。囲まれた文字列データの中身には、開始タグと終了タグで囲まれた別の要素を置くことが許される。これによって、再帰的な入れ子構造を表現する。この構造は、論理的に木構造である。入れ子になっている要素を木構造として見る場合は、直接囲まれている要素を子ノード、その要素を直接囲んでいる要素を親ノードとし、同じ親ノードを共有するノードを兄弟のノードとする。

## 【0004】

プログラム言語からこのような構造化文書进行操作する場合、構造化文書の木構造になっているデータを直接プログラム言語の構造体に転記する方式はなかった。

## 【0005】

これまでの技術としては、構造化文書全体を、各要素をノードとする木構造として主記憶装置内に展開した上で、このデータ構造を操作する方法がある。例えば、ワールド・ワイド・ウェブ・コンソーシアム (World Wide Web Consortium) はドキュメント・オブジェクト・モデル (Document Object Model) を勧告しており (<http://www.w3.org/TR/REC-DOM-Level-1/>)、主記憶内に

木構造として展開したデータ構造に対する各種のデータ操作インターフェースを定義している。

【0006】

また、構造化文書を走査するにつれて、イベントを発生させることで、イベント・ドリブンに構造化文書进行操作する方法がある。XML-DEVメーリングリストで開発され、XML文書を扱う上で事実上の業界標準となっているSimple API for XML (<http://www.megginson.com/SAX/>) では、XMLで書かれた構造化文書を走査し、遭遇したタグ等に対して次々にイベントを発生させ、特定のイベント処理ルーチン（コールバック・ルーチンと呼ぶ）に制御を渡すことで、構造化文書の操作を可能にしている。

【0007】

特開平11-242673では、ある構造化情報フォーマットから別の構造化情報フォーマットへ変換するためのオブジェクト指向システムが提案されているが、プログラム言語処理系には関係していない。

【0008】

【発明が解決しようとする課題】

構造化文書を扱うアプリケーションを作成する場合、次に述べるように、ポインタ操作機能や、イベントによって呼び出されるコールバック・ルーチンの登録機能の無いプログラム言語では実現が困難であるという問題があった。

【0009】

上記従来技術のうち、構造化文書を木構造に展開してプログラム言語から操作する方法では、動的なデータ構造を扱うため、プログラム言語の言語仕様として、ポインタの操作機能、又は、オブジェクト指向言語であればオブジェクト参照の操作機能を持っていなければならない。また、構造化文書をイベント・ドリブンに扱う方法では、プログラム言語の言語仕様として、イベントごとに呼び出されるコールバック・ルーチンを設定する手段を提供していなければならない。例えばプログラム言語COBOL (JIS X 3002-1992 電子計算機プログラム言語COBOL) のように、言語仕様としてデータ用のポインタも、

手続きルーチン用のポインタの概念もない言語では、動的データ構造の扱いも、イベント・ドリブンのプログラミングも困難である。

【0010】

本発明は、ポインタ操作や呼び出されるルーチンの登録機能がないプログラム言語でも、構造化文書のデータを扱える機能を提供することにある。

【0011】

【課題を解決するための手段】

上記目的を達成するために、プログラム言語から呼び出して使用するデータ転記処理部を置く。データ転記処理部は、構造化文書の構造を定義した情報、構造化文書の構造に対応するプログラム言語の構造体の定義情報、及び、文書構造定義と構造体定義の対応情報を入力することで、構造化文書の各要素とプログラム言語の構造体の各項目との間の対応関係と型情報に基づいて、データ型変換を含めた各々の転記方法を設定する。

【0012】

構造化文書の内容データをプログラム言語の構造体に転記する要求をアプリケーションプログラムがデータ転記処理部に出すと、構造化文書の内容データを、個々の要素に対して上述のように設定された方法で、プログラム言語の構造体に転記する。また、逆方向のデータ転記要求として、プログラム言語の構造体データを構造化文書に転記する要求をアプリケーションプログラムがデータ転記処理部に出すと、プログラム言語の構造体データを、個々の項目に対して上述のように設定された方法で、構造化文書に転記する。

【0013】

アプリケーションプログラムは、かかるデータ転記処理部とやり取りするだけで、構造化文書のデータを扱うことが可能になる。

尚、企業間システムにおけるデータ構造を変換するプログラムにあって、前述の企業間で共通に用いる構造化文書のデータ構造を定めておき、前述のプログラムは、前述の企業間で用いる構造化文書のデータ構造を定義した情報と、企業内で用いるプログラム言語の構造体の定義情報と、前述の企業間で用いる構造化文書のデータ構造と前記企業内で用いるプログラム言語の構造体との対応情報と

を取得し、前述の取得した情報に基づいて、前述のプログラム言語の構造体の状態を格納しておくフラグ構造体と前述のフラグ構造体に対応するプログラム言語の構造体を作成し、前述の作成したフラグ構造体と前述のフラグ構造体に対応するプログラム言語の構造体とを用いることでデータ構造を変換することを可能とする。

#### 【0014】

##### 【発明の実施の形態】

以下、本発明の実施の形態を詳細に説明する。

図1は、本発明における構造化文書を扱うアプリケーションプログラムの論理的なシステム構成を示すものである。図2は、図1のシステム構成のうちデータ転記処理部5の一部として、定義情報解析に基づいてあらかじめ構造化文書アクセスルーチン群503を生成する部分を示すものである。図3は、図2で生成した構造化文書アクセスルーチン群503をアプリケーションプログラムから呼び出す方法で図1のシステム構成を実施する形態を示すものである。図4は、構造化文書の構造定義情報としてXMLの文書型を宣言するDTD (Document Type Definition) の例である。図5は、図4の文書構造定義情報と、その定義に従った構造化文書本体の例である。図6は、図4の文書構造定義情報に対応したプログラム言語の構造体定義502の例である。図7は、図6に示す構造体に図5に示す構造化文書本体のデータを転記した場合のデータ設定状態を示すものである。図8は、図4の文書構造定義情報に対応したプログラム言語の構造体定義502のもう一つの例である。図9は、構造化文書アクセスルーチン群503の機能構成を示すものである。図10は、構造化文書の文書構造定義情報1に省略可能要素や繰り返し要素がある場合に、省略の有無や繰り返し数を保持する構造体の例である。図11は、文書構造定義情報1の要素に属性が付く場合、属性値を要素の内容と共に保持する構造体の例である。図12は、図1のアプリケーションプログラム4が、データ転記処理部5を使って構造化文書6を読み込む処理方法のフローチャートである。

#### 【0015】

図1のデータ転記システムの構成において、データ転記処理部5は、構造化文

書の文書構造定義情報1、プログラム言語の構造体定義情報2、及び、文書構造定義と構造体定義との対応情報3を取得し、アプリケーションプログラム4で使用する構造体と構造化文書6との間のデータ転記処理方法確立する。アプリケーションプログラム4は、データ転記処理部5を適宜呼び出すことで構造化文書6を読み書きするというデータ転記方法を用いる。データ転記処理部5は、明示的な呼び出し構文によって呼び出すのではなく、プログラム言語に備わった入出力用の構文の実行で呼び出すように実施しても良い。

## 【0016】

データ転記処理部5は、アプリケーションプログラム4の実行時に、構造化文書の文書構造定義情報1、プログラム言語の構造体定義情報2、及び、文書構造定義と構造体定義との対応情報3を取得する方法で実施しても良いし、アプリケーションプログラム4の実行前に、これら情報を取得して確立したデータ転記方法に従うルーチンを生成しておき、アプリケーションプログラム4が生成されたルーチンを使う方法で実施してもよい。図2以降は、データ転記方法の確立を、アプリケーションプログラムの実行の前の段階であらかじめ行う方法で本発明を実施する形態を説明する。

## 【0017】

図2において、定義情報解析部501は、構造化文書の文書構造定義情報1、プログラム言語の構造体定義情報2、及び、文書構造定義と構造体定義との対応情報3を入力する。ここで、構造化文書の構造定義1は、例えば、XML文書であればDTDである。プログラム言語の構造体定義情報2は、ソースプログラムの構造体変数宣言部分、ソフトウェア開発の上流ツールにあるリポジトリ情報、又は、変数宣言部分を翻訳したコンパイラの間接情報等を使用する。文書構造定義と構造体定義との対応情報3は、構造化文書の各要素からプログラム言語の構造体の各項目への対応を、繰り返し要素、選択可能要素、省略可能要素等を含めて表現する。さらに、構造化文書に繰り返し要素数の最大値の定めが無い場合、最大繰り返し数を指定する。また、構造化文書に文字列長の最大値の定めが無い場合、要素の最大長も指定する。

## 【0018】

次に、定義情報解析部 501 は、文書構造定義と構造体定義との対応付け機能 5011 によって両者を対応付ける。さらに、構造体定義生成機能 5012 によって、プログラム言語の構造体定義 502 を生成する。ここで、プログラム言語の構造体定義情報 2 がソースプログラムの構造体変数宣言部分である場合は、プログラム言語の構造体定義 502 と同一であり、構造体定義生成機能 5012 は不要である。また、プログラム言語の構造体定義情報 2 から、ツール等でプログラム言語の構造体定義 502 相当が生成される場合も構造体定義生成機能 5012 は不要である。

#### 【0019】

定義情報解析部 501 は、次に、データ転記処理生成機能 5013 によって、構造化文書のアクセスルーチン群 503 を生成する。ここで、構造化文書のアクセスルーチン群 503 は、文書構造定義と構造体定義との対応情報 3 に従って、データを転記するためのルーチン群であり、更に、プログラム言語の構造体定義情報 2 に従って、文字列から数値への変換等のデータ型変換も行う。

#### 【0020】

文書構造定義と構造体定義との対応情報 3 は、構造化文書の文書構造定義情報 1 と一緒に持たせることも、プログラム言語の構造体定義情報 2 と一緒に持たせることもできる。

#### 【0021】

構造化文書の文書構造定義情報 1 に定義される文書構造のすべて、又は、ある要素部分の構造のすべてを、その木構造を変えずにプログラム言語の構造体に対応させる場合、構造化文書の木構造からプログラム言語の構造体の構造を一意に決められるため、木構造の形に関する対応情報は必要ない。したがって、文書構造定義と構造体定義との対応情報 3 に構造化文書の各要素に対応する型情報を持たせれば、プログラム言語の構造体定義情報 2 は不要である。

#### 【0022】

これとは反対に、プログラム言語の構造体定義情報 2 に定義されるデータ項目のすべて、又は、構造体の一部分の構造のすべてを、その構造を変えずに構造化文書に対応させる場合、プログラム言語の構造体の構造から構造化文書の木構造

を一意に決められるため、構造化文書の文書構造定義情報1は不要である。この場合、定義情報解析部501の構造体定義生成機能5012は、文書構造定義情報を生成する機能に置き換えることで、構造化文書の文書構造定義情報1を生成することができる。

#### 【0023】

図3において、アプリケーションプログラム4は、図2の定義情報解析部501で生成された構造化文書のアクセスルーチン群503を適宜呼び出すことで構造化文書6を読み書きする。このとき、アプリケーションプログラム4は、図2の定義情報解析部501で生成されたプログラム言語の構造体定義502を取り込んで、読み書きのデータのやり取りに使用する。

#### 【0024】

図4は、構造化文書の文書構造定義情報1の例として、XMLのDTDの例を示している。要素`order`の宣言71では、要素`order`は、要素名が、この文書型の名前と共通であり、文書型のルート要素を示している。また、要素`order`は、その内側に、一つの`address`要素と一つ以上の`item`要素を、この順に持つことを定義している。`address`要素は宣言72で、`item`要素は宣言73で、それぞれ宣言されている。

#### 【0025】

アクセスルーチン群503では、アクセスの単位を、構造化文書6全体にすることもできるし、一部分にすることもできる。図4で定義される文書型の構造化文書へのアクセス用にアクセスルーチン群503を生成する場合、構造化文書6にアクセスする単位として、`order`要素を選択すれば、`order`は文書型のルート要素なので、アプリケーションプログラム4は、文書全体を一度にアクセスすることになる。一方、アクセス単位を、`address`要素と`item`要素に分割してアクセスルーチン群503を生成すると、アプリケーションプログラム4は、まず、`address`要素にアクセスし、次に`item`要素に1度以上繰り返してアクセスすることで、構造化文書6全体をアクセスすることができる。アクセス単位の指定は、プログラム言語の構造体定義情報2、又は、文書構造定義と構造体定義との対応情報3で行う。

## 【0026】

図5では、図4のDTDに従ったXML文書本体の例を図4のDTDと共に示している。アクセスルーチン群503のアクセス単位を宣言71で宣言されるorder要素とする場合は、構造化文書本体(710)を一度にアクセスする。また、アクセスルーチン群503のアクセス単位を宣言72で宣言されるaddress要素と宣言73で宣言されるitem要素に分割する場合、XML文書のaddress要素(720)へのアクセスと、XML文書のitem要素(731、732)への、2回のアクセスで、文書全体をアクセスする。

## 【0027】

図6は、プログラム言語の構造体定義502の例であり、図4に示すDTDに対応して、宣言71で宣言されるorder要素をアクセス単位として生成されるプログラム言語COBOLの構造体定義(5021)を示す。ここでは、図4で宣言される要素名order、address、item、itemname、price、及び、quantityに、図6に示すCOBOLのデータ項目名ORDER、ADDRESS、ITEM、ITEMNAME、PRICE、及び、QUANTITYをそれぞれ対応させている。データ項目ADDRESSは20文字の英数字項目に、データ項目ITEMNAMEは15文字の英数字項目に、データ項目PRICE及びQUANTITYは十進数9けたの数字項目に、そして、データ項目ITEMの繰り返し数は2回に、それぞれ宣言している。このように、図2において、構造化文書の文書定義情報1に指定されないが、プログラム言語の構造体定義502では必要になる長さ情報、型情報、及び、繰り返し項目の最大繰り返し数等の情報は、プログラム言語の構造体定義情報2、又は、文書構造定義と構造体定義との対応情報3に設定しておく。

## 【0028】

図7は、図5に示す構造化文書本体(710)の各要素のデータを、図6に示すプログラム言語COBOLの構造体定義(5021)の各データ項目に転記した場合のデータの設定状態(5022)を示す。図7の各データ項目名は図6の同名のデータ項目名に対応する。図5のitem要素の2回の繰り返し(731、732)に対して、データ項目ITEMが2回繰り返され、それぞれに対応す

る要素データが設定されている。

#### 【0029】

図8は、プログラム言語の構造体定義502のもう一つの例として、図4のDTDに対応して2つの構造体定義が含まれる例を示す。一つは宣言72で宣言される`address`要素をアクセス単位として生成されるCOBOLの構造体定義(5023)であり(ただし、`address`要素は末端の要素で内側に構造を持たないため、COBOLの定義もここでは構造を持たない基本型のデータ項目になっている)、もう一つは、宣言73で宣言される`item`要素をアクセス単位として生成されるCOBOLの構造体定義(5024)である。ここでは、図5で宣言される要素名`address`、`item`、`itemname`、`price`、及び、`quantity`に、図8に示すデータ項目名ADDRESS、ITEM、ITEMNAME、PRICE、及び、QUANTITYをそれぞれ対応させている。図6の構造体定義(5021)とは異なり、`item`要素の繰り返しは指定しない。その代わりに、アプリケーションプログラム4の中で`item`要素のアクセスを繰り返すことで複数の`item`要素にアクセスする。

#### 【0030】

図9において、構造化文書アクセスルーチン群503は、初期化処理機能5031、構造化文書アクセス単位の読み書き機能5032、構造化文書内位置付け機能5035、及び、終了処理機能5036からなる。構造化文書アクセス単位の読み書き機能5032は、構造化文書から構造体への読み込み機能5033と構造体から構造化文書への書き出し機能5034からなる。

#### 【0031】

構造化文書アクセス単位の読み書き機能5032は、アクセス単位毎に一つ生成される。したがって、一つの文書型に複数のアクセス単位を設定する場合は、複数の構造化文書アクセス単位の読み書き機能5032が生成される。構造化文書アクセス単位の読み書き機能5032を実装するルーチンの入り口点は、アクセス単位に一そろいずつ生成しても良いし、一そろいだけ生成して、呼び出し時に引数やその他の方法で与えられる情報によってアクセス単位を選択しても良い。

## 【0032】

初期化処理機能5031は、構造化文書6を読み込む場合には、まず、指定された場所（ファイル名や主記憶装置の番地）に格納されている構造化文書をすべて読み込み、構造化文書の構造を、主記憶装置内で木構造のデータ構造に展開する。構造化文書6を書き出す場合には、指定された書き出し場所（ファイル名や主記憶装置の番地）を確保又は登録する。読み込みの場合も書き出しの場合も、構造化文書の格納場所にファイルを指定するなら、ファイルを先にオープンしておく。構造化文書6に対して読み込みと書き出しを行う場合には、上述の読み込み用の初期化、及び、書き出し用の初期化を行う。

## 【0033】

構造化文書アクセス単位の読み書き機能5032は、状態として、アクセス対象の構造化文書内の現在位置を保持する。初期状態は、構造化文書の先頭である。

## 【0034】

構造化文書から構造体への読み込み機能5033は、主記憶装置内で木構造に展開した構造化文書から、アクセス単位分の要素データを与えられた構造体の対応するデータ項目に読み込むことでデータ転記を実現する。実施の形態として、現在位置から読み込む方法と、対応するアクセス単位を、現在位置から構造化文書の前又は後ろ方向に検索し、見つかったアクセス単位を読み込む方法がある。ここで、読み込んだ要素の終了位置を、現在位置情報として保持する。次に、構造化文書から構造体への読み込み機能5033のいずれかが呼ばれたときには、現在位置から先にあるアクセス単位を探して、与えられた構造体を読み込む。構造化文書から構造体への読み込み機能5033が呼ばれたとき、対応するアクセス単位が構造化文書内に無ければ、エラーコードを返却することで、アプリケーションプログラム4に次の操作を選択する機会を与える。

## 【0035】

構造体から構造化文書への書き出し機能5034は、引数やその他の方法で渡される構造体データを、呼び出されるたびに直接構造化文書の文字列として書き出すことでデータ転記を実現する。実施の形態として、現在位置から書き出す方

法と、対応するアクセス単位が存在可能な位置を、構造化文書の現在位置から後ろ方向に検索し、見つかった位置にアクセス単位を書き出す方法がある。構造体から構造化文書への書き出し機能 5034 は、与えられたアクセス単位の構造体に格納されたデータを次のように処理することで構造化文書の文字列を生成する。データ項目は構造体型又は基本型のどちらかであり、構造体型は、内部にデータ項目（構造体型又は基本型）を一つ以上持つという構造になっている。まず、データ項目が基本型であれば、型に従ってデータ値を表す文字列に変換し、構造体の文書構造定義と構造体定義との対応情報 3 で指定された要素の開始タグと終了タグで、変換後の文字列を囲む。それが構造体型であれば、その構造体を含むデータ項目をすべて開始タグと終了タグで囲んでから、それら全体を、構造体の文書構造定義と構造体定義との対応情報 3 で指定された要素の開始タグと終了タグで囲む。なお、構造体から構造化文書への書き出し機能 5034 には、引数やその他の方法で呼び出し時に情報を与えることで選択可能なオプションとして、データ値を表す文字列の前、又は、後ろの余分な空白を削除する機能を入れることができる。

#### 【0036】

上述の手順だけでは、図 8 のように構造化文書のアクセス単位を分割した場合に、構造体 5023 を書き出しても構造体 5024 を書き出しても、全体を囲んでいる `order` 要素の開始タグと終了タグを付与しない。最外側の開始タグ及び終了タグ書き出しでなくとも、二つの部分的アクセス単位の書き出しの間でも、同様にタグを付与しない場合がある。そこで、構造体から構造化文書への書き出し機能 5034 には、部分的なアクセス単位を書き出すときに、その前に出現していなければならない開始タグ及び終了タグを自動的に付与する機能を持たせる。また、終了処理機能 5036 には、構造化文書 6 が終了する前に閉じていなければならないタグを付与する機能を持たせる。

#### 【0037】

構造体から構造化文書への書き出し機能 5034 のもう一つの実施形態として、各呼び出しでは、引数やその他の方法で渡される構造体データを構造化文書の部分的な木構造表現に変換し、主記憶装置内で木構造表現にした構造化文書に付

加していき、終了処理機能 5036 が呼ばれるときに、木構造表現全体から文字列表現の構造化文書を生成する実施形態がある。この実施形態でも、現在位置から書き出す方法と、対応するアクセス単位が存在可能な位置を、構造化文書の現在位置から前又は後ろ方向に検索し、見つかった位置にアクセス単位を書き出す方法がある。例えば XML 文書の場合、ドキュメント・オブジェクト・モデル形式の木構造を作成し、終了処理機能 5036 は、現在のドキュメント・オブジェクト・モデル形式木構造に基づいて、文字列の XML 文書を生成する。この実施形態では、アプリケーションプログラム 4 は、構造化文書 6 の中で変更しない部分をアクセス単位にする必要が無いいため、既存の構造化文書の一部分だけを修正するアプリケーションに向いている。

## 【0038】

構造化文書内位置付け機能 5035 は、構造化文書内の位置を表す情報を受け取り、その位置に新たに現在位置を設定する。構造化文書内の位置を表す方法については、XML であれば、XML Path Language (<http://www.w3.org/TR/1999/REC-xpath-19991116>) を用いることができる。

## 【0039】

終了処理機能 5036 は、ファイルのクローズ処理、メモリの解放処理、閉じていないタグの付与処理等を行う。また、構造体から構造化文書への書き出し機能 5034 が主記憶装置内に木構造表現で構築した構造化文書を、終了処理機能 5036 が文字列表現の構造化文書を生成する実施形態の場合は、この生成処理も行う。

## 【0040】

図 10 において、XML の部分的な DTD (81) は、要素 A の内側に、要素 B1 又は B2 が存在し、その次に要素 C が存在しても良いが省略可能で、最後に要素 D が一つ以上繰り返すという構造を定義している。また、要素 B1、B2、C、及び、D は、それぞれ文字列データと定義している。COBOL の部分的な構造体 (82) は、部分的な DTD (81) に対応する構造体の例であり、データ項目 A、B1、B2、C、及び、D は、それぞれ要素 A、B1、B2、C、及

び、Dに対応する。データ項目B 1、B 2、C、及び、Dは、長さ20文字の英数字項目に、データ項目Dの繰り返し数は10回に、それぞれ固定値で定義し、これらのデータ項目を含む構造体としてデータ項目Aを定義している。ここで、選択要素である要素B 1と要素B 2のそれぞれに独立したデータ項目を与えているが、データ項目B 2をデータ項目B 1の再定義項目としてデータ領域を共用させても良い。

## 【0041】

構造化文書のアクセス単位を要素Aとする場合、上述の方法だけでは、構造化文書の読み込み時、要素B 1又はB 2のどちらが存在したか、要素Cは存在したかどうか、そして、要素Dの出現回数はいくつであったか、それぞれアプリケーションプログラムでは確認できない。また、書き出し時には、要素B 1又はB 2のどちらを書き出すのか、要素Cは書き出すのか書き出さないのか、要素Dは何回書き出したら良いのかをアプリケーションプログラムから指定できない。そこで、図2の構造体定義生成機能5012に、要素の存在の有無、及び、出現回数を納めるフラグ構造体も、プログラム言語の構造体定義502の一部として生成する機能を加える。また、構造化文書から構造体への読み込み機能5033には、フラグ構造体に値を設定する機能を追加し、構造体から構造化文書への書き出し機能5034には、フラグ構造体に設定された値に従ってアクセス単位の構造化文書を書き出す機能を追加する。（なお、フラグ構造体とは、構造体の状態を保持するものである。たとえば、データの有無や、データの長さ、データの型、データの出現回数などである。詳しくは以下で図10の(83)を用いて説明する。）

COBOLの部分的な構造体(83)は、部分的なDTD(81)及びCOBOLの部分的な構造体(82)に対応するフラグ構造体の例である。要素B 1、B 2、及び、Cに対応して、データ項目B 1、B 2、及び、Cが一文字のデータ値を持ち、要素の存在の有無を表現する。また、要素Dに対応して、データ項目Dが十進数9けたのデータ値を持ち、繰り返し要素の出現回数を表現する。

## 【0042】

フラグ構造体は、構造化文書の要素データに対応する構造体データ項目上の開

始位置と長さの情報を表現する目的でも使用できる。構造化文書から構造体への読み込み機能 5033 が、構造化文書の要素データの文字列を構造体の固定長データ項目に転記すると、なんらかの埋め草文字をデータの前又は後ろに挿入する。このままでは、アプリケーションプログラム 4 からは、そもそも構造化文書上でデータであった部分と埋め草として埋め込まれた文字との区別がつかない。データ項目に対応して、実際に読み込んだデータを格納した開始位置と長さの情報を、フラグ構造体に持たせることで、アプリケーションプログラム 4 は、実際に読み込まれたデータ部分を認識できる。また、構造体から構造化文書への書き出し機能 5034 が、構造体のデータ項目に格納された文字列データを構造化文書の要素に転記するとき、対応するフラグ構造体に格納された開始位置と長さ情報に従って部分的に転記することで、アプリケーションプログラム 4 は、転記するデータ部分を制御できる。

#### 【0043】

さらに、フラグ構造体は、構造化文書のデータが、構造体として用意しているデータ領域に収まり切らない状態を表現する目的でも使用できる。構造化文書から構造体への読み込み機能 5033 が構造化文書を読み込むとき、要素データ長や要素の繰り返し数が、対応する構造体側で固定に定義しているデータの長さや繰り返し数を超えているとき、あふれる分のデータは転記できない。繰り返し数がデータ構造で用意した数を超えたことを表すフラグ、及び、長さがデータ構造で用意した数を超えたことを表すフラグを、フラグ構造体に付加し、さらに、実際のデータの繰り返し数情報データ項目、及び、実際のデータの長さ情報データ項目を付加し、構造化文書から構造体への読み込み機能 5033 がこれらの情報を設定することで、アプリケーションプログラム 4 は、読み込みエラー状態を認識できると共に、エラーに対処する処理を実行することができる。

#### 【0044】

なお、フラグ構造体を使用せずに、構造体に含まれる各基本型データ項目に省略時に設定するそれぞれの値をあらかじめ文書構造定義と構造体定義との対応情報 3 で指定しておき、これを構造化文書アクセス単位の読み書き機能 5032 が利用することで、要素の存在の有無を表現する実施の形態もある。構造化文書か

ら構造体への読み込み機能 5 0 3 3 は、構造化文書 6 の内容データを構造体読み込むと共に、選択されなかった選択可能要素、及び、存在しなかった省略可能要素に対応するデータ項目に、並びに、繰り返し可能要素の構造化文書内での実際の繰り返し数が、構造体で用意していた繰り返し数より少ないとき、データが読み込まれなかったデータ項目に、省略時設定値を設定する。反対に、構造体から構造化文書の書き出し機能 5 0 3 4 は、省略時設定値が設定されているデータ項目の対応する要素を書き出さない。アプリケーションプログラム 4 は、読み込みのとき、省略時設定値が設定されたデータ項目に対応する要素が構造化文書 6 に存在しなかったことを認識でき、書き出しのとき、データ項目にあらかじめ省略時設定値を設定しておくことで、書き出さない要素を指定できる。

#### 【 0 0 4 5 】

この実施の形態では、フラグ構造体を用いる実施形態に比較して、機能的にいくつかの制限がある。基本型のデータ項目でなく構造体型の項目レベルでの要素の省略を表現できないし、省略時設定値を意味のあるデータから除外する必要がある。また、構造化文書内の繰り返し数が構造体で用意した数を超えたことを表現する手段を別に用意する必要がある。選択可能要素に対応して用意するデータ項目は領域を共用せずに、それぞれに独立したデータ項目として用意し、それぞれの省略時設定値を判断できるようにする必要がある。しかしながら、本来のデータを扱う構造体とは別の構造体を用意する必要が無いので、データの扱いは単純になる。

#### 【 0 0 4 6 】

図 1 1 において、部分的な DTD ( 9 1 ) には、要素 `room` と、要素 `room` に付属する属性 `smoking` が定義されている。部分的な XML 文書 ( 9 2 ) は、部分的な DTD ( 9 1 ) に従った XML 文書の例であり、要素 `room` の内容データには 0 3 0 8、属性 `smoking` には "no" が設定されている。開始タグと終了タグで囲まる要素の内容データと共に、要素の属性値をプログラム言語の構造体で扱うには、構造化文書の要素に対応して、構造体の中に、属性値用のデータ項目と要素の内容用のデータ項目を持つ構造体を定義する。部分的な COBOL の構造体定義 ( 9 3 ) は、部分的な DTD ( 9 1 ) に対して、属性値

と要素内容の両方を表現させたCOBOLの構造体の例である。要素roomの値は、COBOLでは、構造体ROOMの中のデータ項目VALに対応付けられ、要素roomの属性smokingの値は、COBOLでは、構造体ROOMの中の構造体ATTLISTの中のデータ項目SMOKINGに対応付けられている。

## 【0047】

構造化文書の文書構造定義情報1に属性情報が含まれる場合、プログラム言語の構造体定義情報2に、属性情報を格納するデータ項目を定義すると共に、文書構造定義と構造体定義との対応情報3で、両者の対応を指定する。さらに、データ転記処理部生成機能5013では、属性情報の転記処理を含めた構造化文書のアクセスルーチン群503を生成する。

## 【0048】

図12は、アプリケーションプログラム4が、構造化文書6を読み込む処理のフローチャートである。構造化文書のアクセスルーチン群503があらかじめ生成されており、これらのルーチンを呼び出すことで、構造化文書6を読み込む。まず、初期化処理機能5031を呼び、構造化文書の格納されたファイルのオープン処理や構造化文書の主記憶装置内での木構造への展開等の必要な初期化処理を行う（ステップ41）。次に、構造化文書から構造体への読み込み機能5033を呼び、この時、対応する構造体を引数やその他の方法で渡し、定義情報解析部501に対してあらかじめ設定したアクセス単位分の構造化文書データを構造体取得する（ステップ42）。次に、アクセス単位の繰り返しが終わったかどうかを、構造化文書から構造体への読み込み機能5033の結果で判断し（ステップ43）、繰り返しが続く場合はステップ42に戻り、終了している場合はステップ44に行ってこの処理を終了する。構造化文書6の全体を一度に読み込む場合は、ステップ43の判断による繰り返し処理は不要である。構造化文書6を書き出す場合は、ステップ42で、構造化文書から構造体への読み込み機能5033の代わりに構造体から構造化文書への書き出し機能5034を呼べば良い。このように構造化文書のアクセス処理は単純化される。

## 【0049】

以上述べたように、本発明によれば、構造化文書のアクセスルーチン群を呼ぶだけで、構造化文書とプログラム言語の構造体との間で内容データを一度に転記できるため、構造化文書の内容のアクセスに、動的にデータ構造を構築したり、そのデータ構造を渡り歩くためにポインタ操作をしたり、状態管理しながらイベント処理のコール・バックルーチンを駆使したりする必要がなくなる。そのため、プログラミングの負担を減らすことができると共に、プログラム不良を作り込む機会を少なくすることができる。また、特に、COBOL等のポインタの概念を持たない言語でも、構造化文書の処理が可能になる。

#### 【 0 0 5 0 】

図 1 3 を用いて本発明の他の実施例を説明する。

ここで、ネットワークを介して接続されている企業システム A (1300)、企業システム B (1302)、企業システム C (1304) の間で顧客データをやり取りする場合を示す。

#### 【 0 0 5 1 】

予め、各企業間でデータを交換する場合のデータ構造 (1306) を定めておくものとする。

各企業内では、各企業固有のアプリケーションおよびデータ構造を用いるものとする。

たとえば、企業システム A で用いるデータの構造体は (1301) であり、A 社内でのデータ構造 (1306) と各社共通のデータ構造 (1306) とのデータ構造を変換するためのプログラム (1307) を A 社の企業システムが有しているものとする。尚、図示していないが、各企業システムには、業務に必要なハードウェア・ソフトウェアを有しているものとする。

企業システム B も同様に、B 社内で用いるデータ構造 (1303) と各社共通で用いるデータ構造 (1306) とを変換するためのプログラム (1308) を有しているものとする。

企業システム C も同様に、C 社内で用いるデータ構造 (1305) と各社共通で用いるデータ構造 (1306) とを変換するためのプログラム (1309) を有しているものとする。

る。

【 0 0 5 2 】

各社のシステムが有するデータ変換プログラム(1307,1308,1309)は、本発明の図3で示した構成でも良いし、他のものでも良い。尚、データ変換プログラム(1307,1308,1309)は、アプリケーションプログラム(4)と、プログラム言語の構造型定義(502)と、構造化文書のアクセスルーチン群(503)とを含む一つのプログラムとして図示したが、この構成以外で本発明を実施しても良い。

【 0 0 5 3 】

上述のように、予め企業間で送受信する構造化文書のデータ構造を定めておき、各企業内で用いるデータ構造との変換プログラムを各企業システムで有することにより、企業間でデータ交換を行う際に、各企業内システムで従来より用いていたデータ構造を変更することなく、他企業システムとのデータ交換を可能とする。企業内システムにおいては、従来から用いているデータ構造に変更を及ぼさずにすむため、企業内の従来から用いているアプリケーションの変更やデータベース管理にかかる手間が省け、システム変更・構築にかかる労力・時間を軽減できる。また、新たなデータ構造でデータを交換することになっても、従来からのデータ資源・ソフトウェア資源に及ぼす影響を少なくすることができる。

【 0 0 5 4 】

たとえば、各企業内システムにおいてはCOBOLのアプリケーションを用いており、各企業内では別々のデータ構造でデータを管理しており、企業間で送受信するデータ構造をXMLなどの構造化文書で定め、上述したプログラムを用いて本発明を適用することも可能である。

【 0 0 5 5 】

図14を用いて本発明の他の実施例を説明する。

集計センタ(1400)、A支店システム(1403)、B支店システム(1404)各々がネットワークで接続されているものとする。

【 0 0 5 6 】

集計センタにおいては、集計センタ内で用いるデータ構造(1401)を用いて、各支店から送信されたデータを処理するものとする。また、集計センタにおいては

、集計センタ内で用いるデータ構造(1401)でデータの処理を行うものとする。

【0057】

A支店システムにおいては、顧客からの注文データをメールプログラムもしくはWebブラウザ等を経由して顧客端末(1405)、携帯電話(1406)などで受けつけ、A支店システム内で処理するものとする。たとえば、A支店システムでは、ポインタを使ってデータを管理しているものとする。

【0058】

B支店システムにおいては、B支店業務クライアントプログラムからの要求をB支店業務サーバプログラムで処理し、配列型のデータ構造でデータを管理しているものとする。

【0059】

集計センタで、A支店、B支店その他各支店のデータを集計するため、各支店ー集計センタ間で用いるデータ構造(1402)を定めておき、各支店は集計センタとの送受信に用いるデータ構造(1402)に基づいてデータを送受信し、集計センタはデータ変換プログラム(1407)を用いて、各支店から受信したデータ構造(1402)を、センタ内で用いるデータ構造(1401)へ変換してデータ集計処理を行う。

【0060】

このように本発明を適用することで、集計センタ内で従来より用いられていたデータ構造を変更することなく、各支店からのデータの処理を行うことができるため、センタ内でのアプリケーションの変更に掛かる時間・労力などが軽減できる。

【0061】

たとえば、集計センタ内で従来からCOBOLのアプリケーションを用いており、集計センター支店間でのデータ交換をXMLなどの構造化文書を用いる場合などに本発明を適用することも可能である。

【0062】

【発明の効果】

以上述べたように、本発明によれば、構造化文書とプログラム言語の構造体との間で内容データを転記できるためプログラム作成の負担を減らすことができる

【図面の簡単な説明】

【図 1】

本発明における構造化文書を扱うアプリケーションプログラムの論理的なシステム構成のブロック図である。

【図 2】

図 1 のシステム構成のうち、データ転記処理部 5 の一部として、定義情報解析に基づいてあらかじめ構造化文書アクセスルーチン群 5 0 3 を生成する部分を示すブロック図である。

【図 3】

図 2 で生成した構造化文書アクセスルーチン群 5 0 3 をアプリケーションプログラムから呼び出す方法で図 1 のシステム構成を実施する形態を示す。

【図 4】

構造化文書の構造定義情報として XML の文書型を宣言する DTD (Document Type Definition) の例である。

【図 5】

図 4 の文書構造定義情報と、その定義に従った構造化文書本体の例である。

【図 6】

図 4 の文書構造定義情報の全体にマッピングしたプログラム言語の構造体定義 5 0 2 の例である。

【図 7】

図 6 に示す構造体に図 5 に示す構造化文書本体のデータを転記した場合のデータ設定状態を示す。

【図 8】

図 4 の文書構造定義情報に対して 2 つに分割してマッピングしたプログラム言語の構造体定義 5 0 2 の例である。

【図 9】

構造化文書アクセスルーチン群 5 0 3 の機能構成を表すブロック図である。

【図 1 0】

構造化文書の文書構造定義情報 1 に省略可能要素や繰り返し要素がある場合に、省略の有無や繰り返し数の情報を保持する構造体の例である。

【図 1 1】

文書構造定義情報 1 の要素に属性が付く場合、属性値を要素の内容と共に保持する構造体の例である。

【図 1 2】

図 1 のアプリケーションプログラム 4 が、データ転記処理部 5 を使って構造化文書 6 を読み込む処理方法のフローチャートである。

【図 1 3】

本発明を企業間システムに適用した例である。

【図 1 4】

本発明を支店－集計センタにおけるシステムに適用した例である。

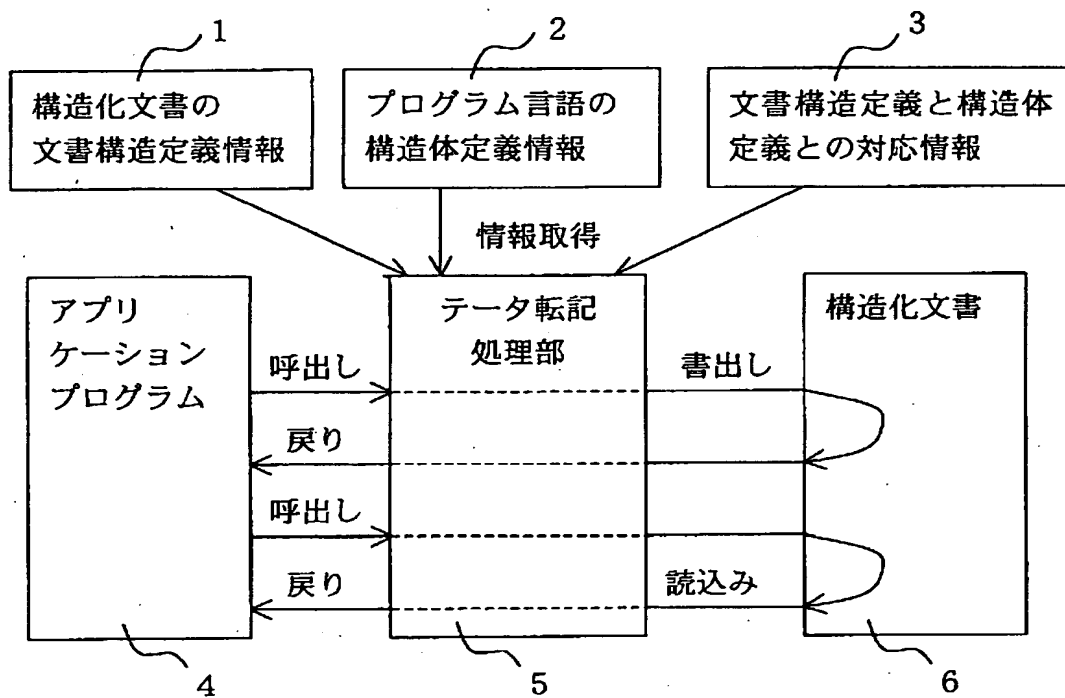
【符号の説明】

- 1 構造化文書の文書構造定義情報
- 2 プログラム言語の構造体定義情報
- 3 文書構造定義と構造体定義との対応情報
- 4 アプリケーションプログラム
- 5 データ転記処理部
- 6 構造化文書

【書類名】 図面

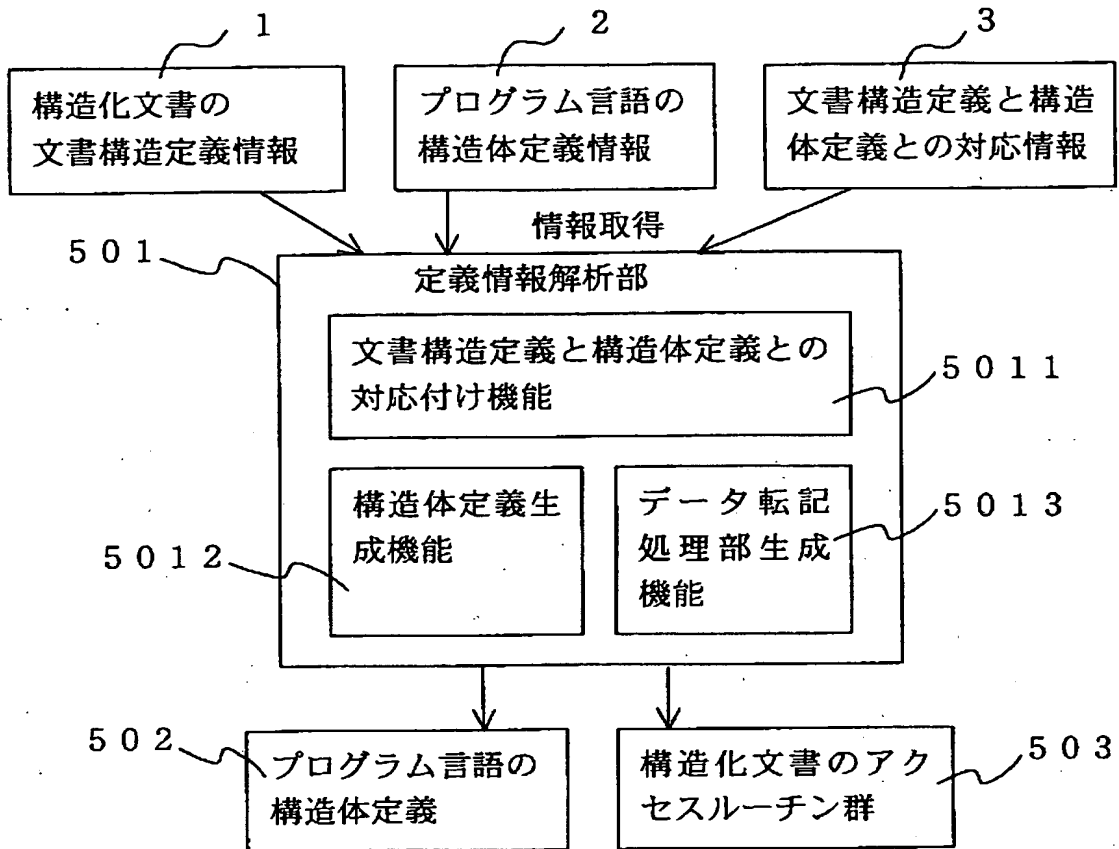
【図 1】

図 1



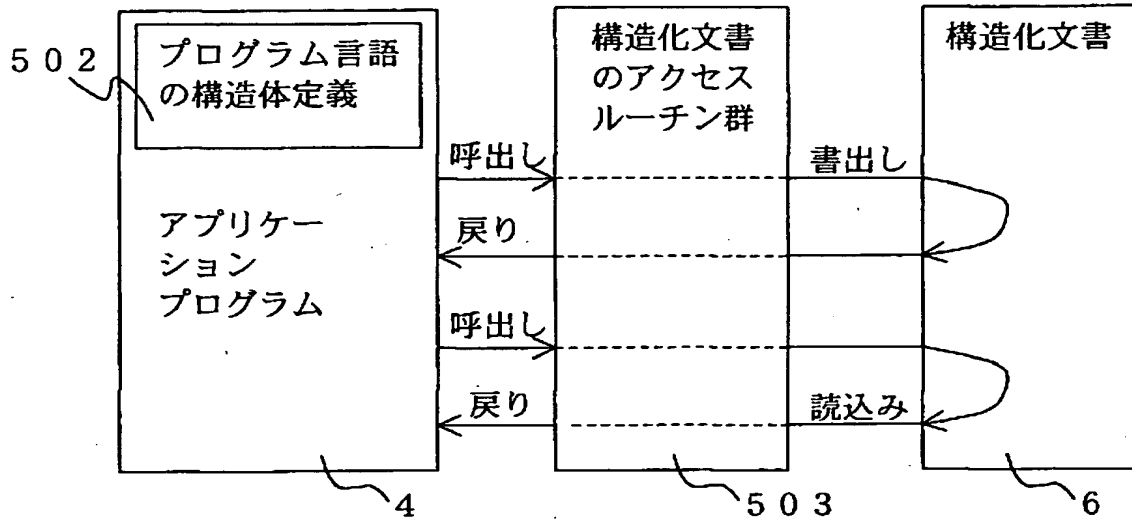
【図 2】

図 2



【図3】

図3



【図4】

図4

```

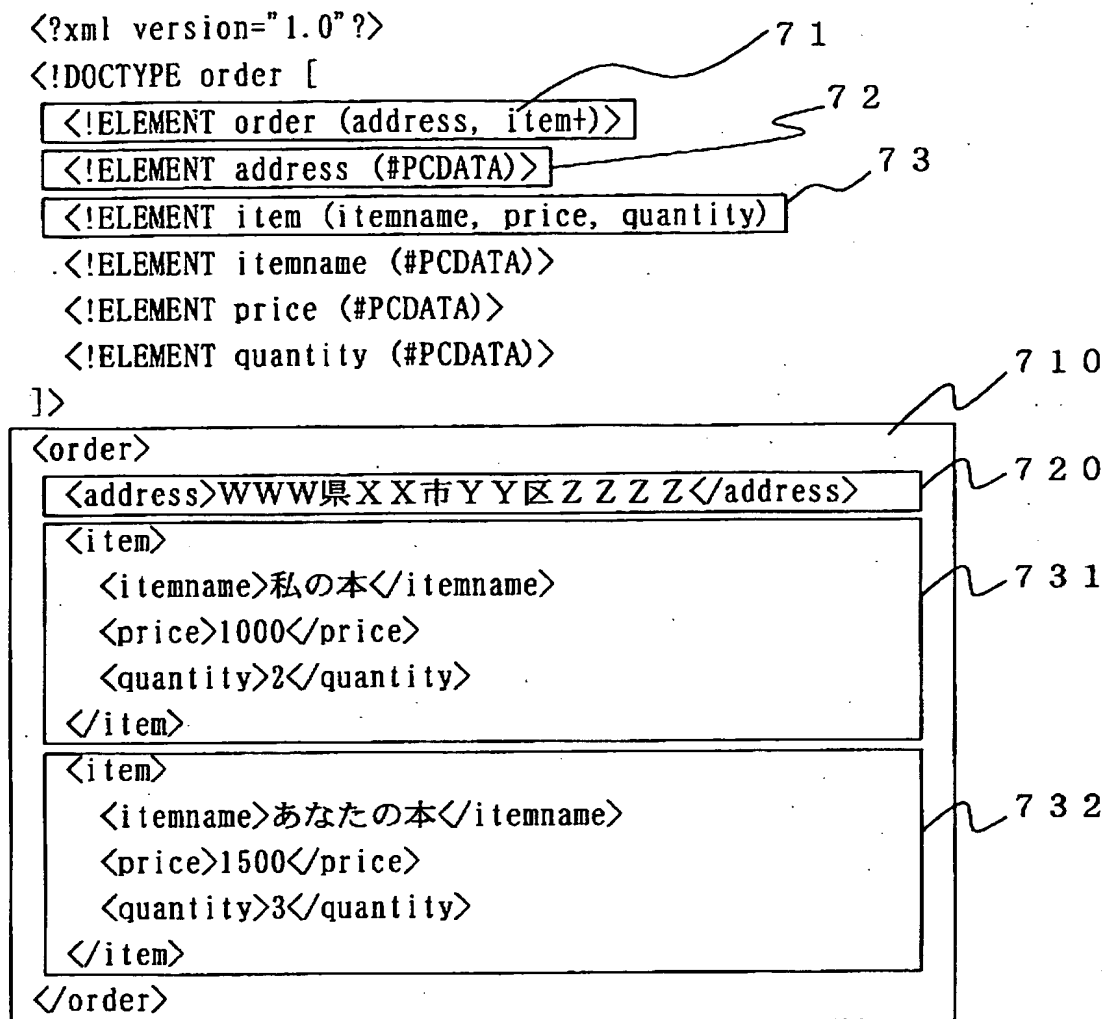
<?xml version="1.0"?>
<!DOCTYPE order [
  <!ELEMENT order (address, item+)>
  <!ELEMENT address (#PCDATA)>
  <!ELEMENT item (itemname, price, quantity)
  <ELEMENT itemname (#PCDATA)>
  <ELEMENT price (#PCDATA)>
  <ELEMENT quantity (#PCDATA)>
]>
  
```

Reference numerals point to specific parts of the XML code:

- 71:** Points to the opening tag of the `<!ELEMENT order (address, item+)>` declaration.
- 72:** Points to the opening tag of the `<!ELEMENT address (#PCDATA)>` declaration.
- 73:** Points to the opening tag of the `<ELEMENT itemname (#PCDATA)>` declaration.

【図5】

図5



【図6】

図6

5021

|    |                         |
|----|-------------------------|
| 01 | ORDER.                  |
| 02 | ADDRESS PIC X(20).      |
| 02 | ITEM OCCURS 2.          |
| 03 | ITEMNAME PIC X(15).     |
| 03 | PRICE PIC 9(9) COMP.    |
| 03 | QUANTITY PIC 9(9) COMP. |

【図7】

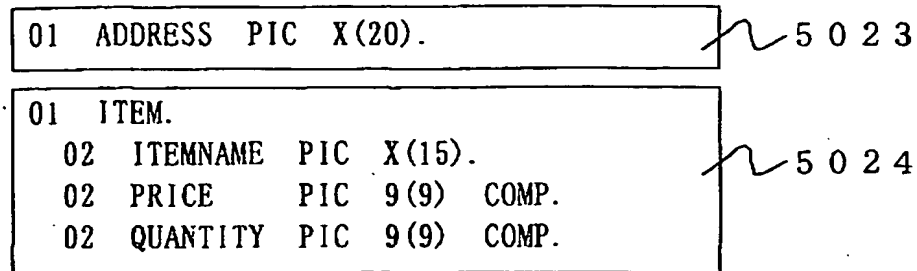
図7

5022

| データ項目名 |          | 値              |
|--------|----------|----------------|
| ORDER  | ADDRESS  | WWW県XX市YY区ZZZZ |
|        | ITEM (1) | ITEMNAME 私の本   |
|        |          | PRICE 1000     |
|        |          | QUANTITY 2     |
|        | ITEM (2) | ITEMNAME あなたの本 |
|        |          | PRICE 1500     |
|        |          | QUANTITY 3     |

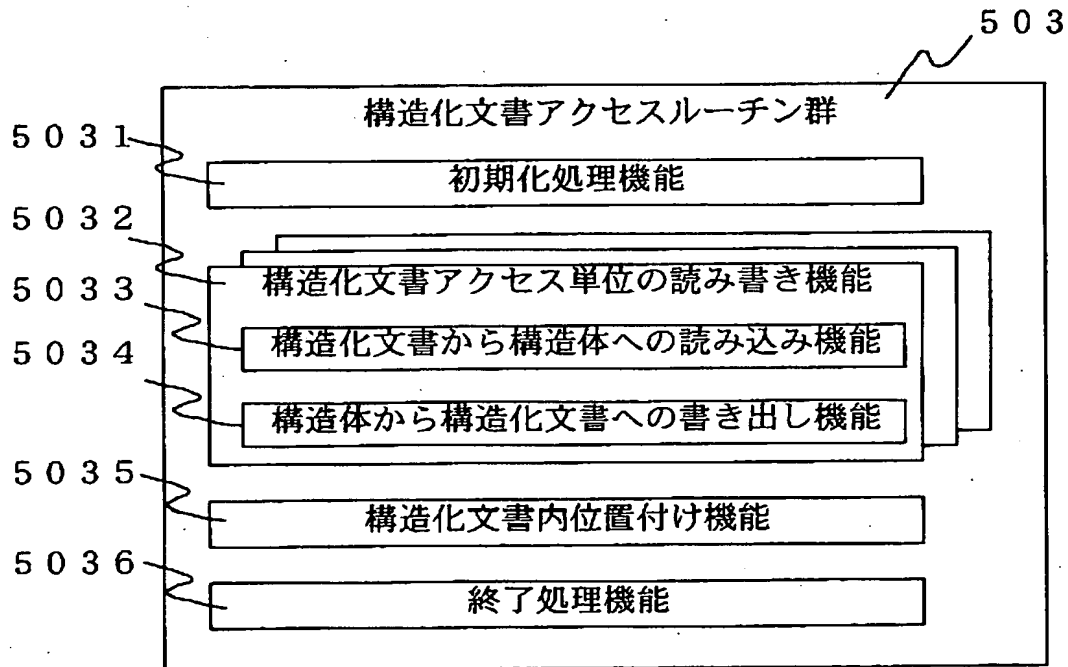
【図 8】

図 8



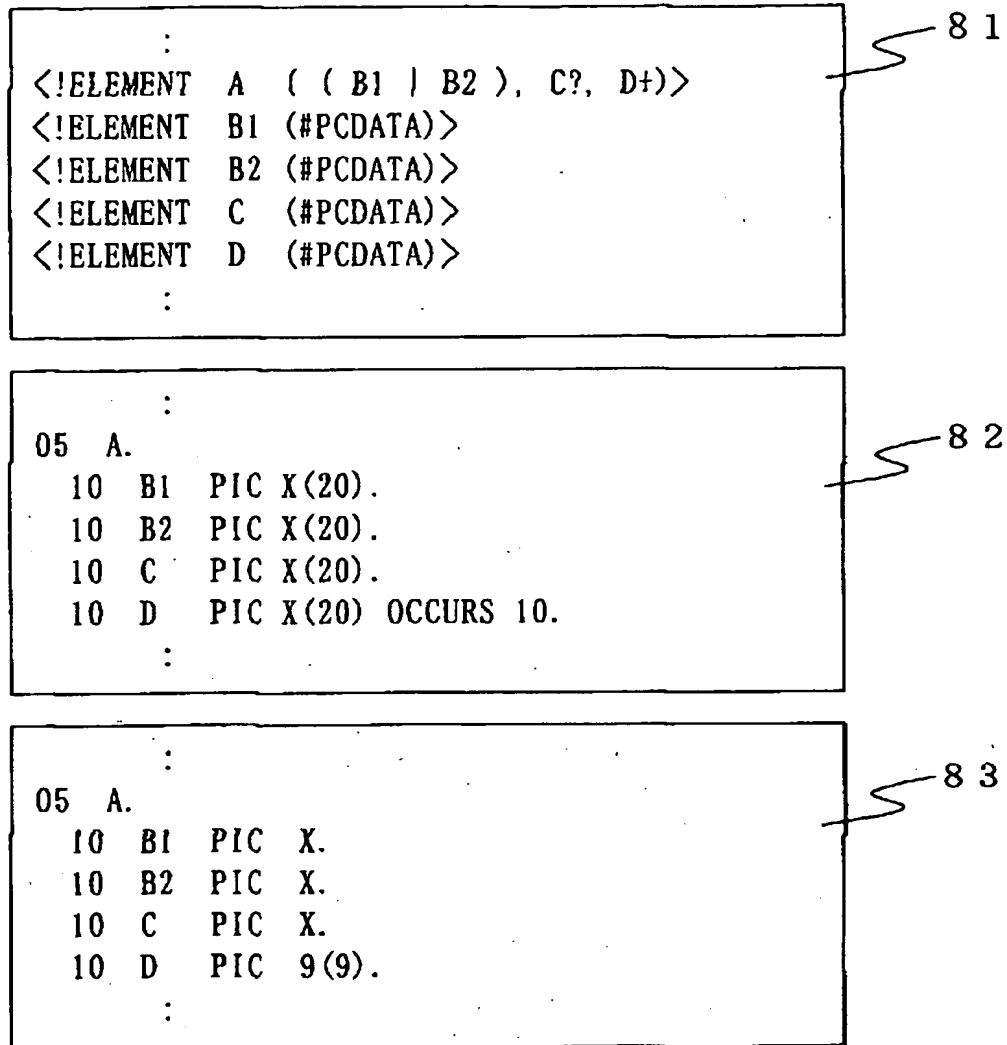
【図 9】

図 9



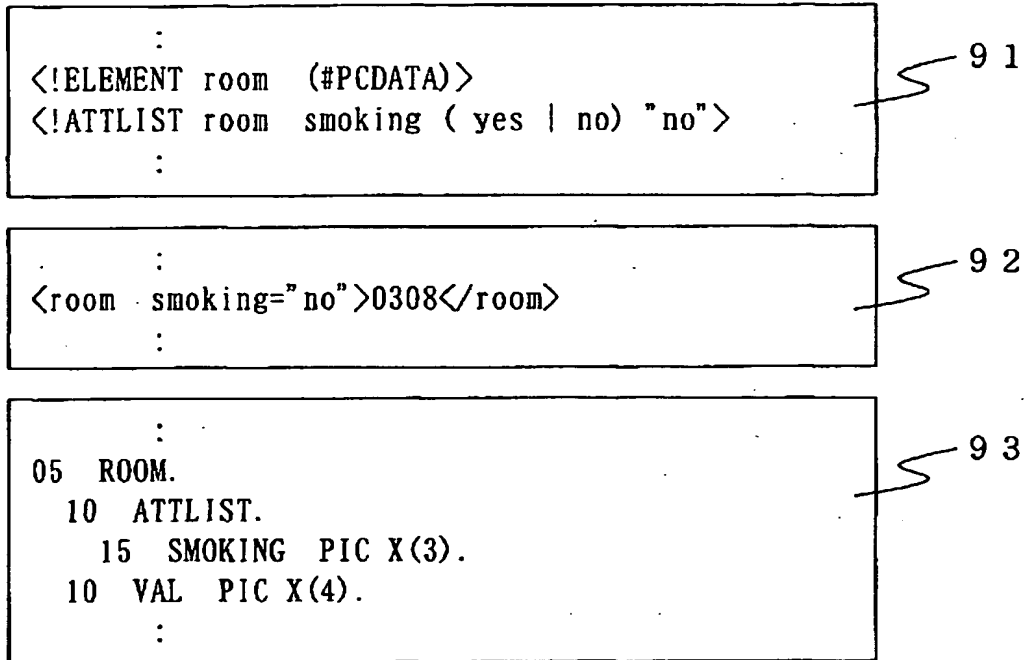
【図10】

図10



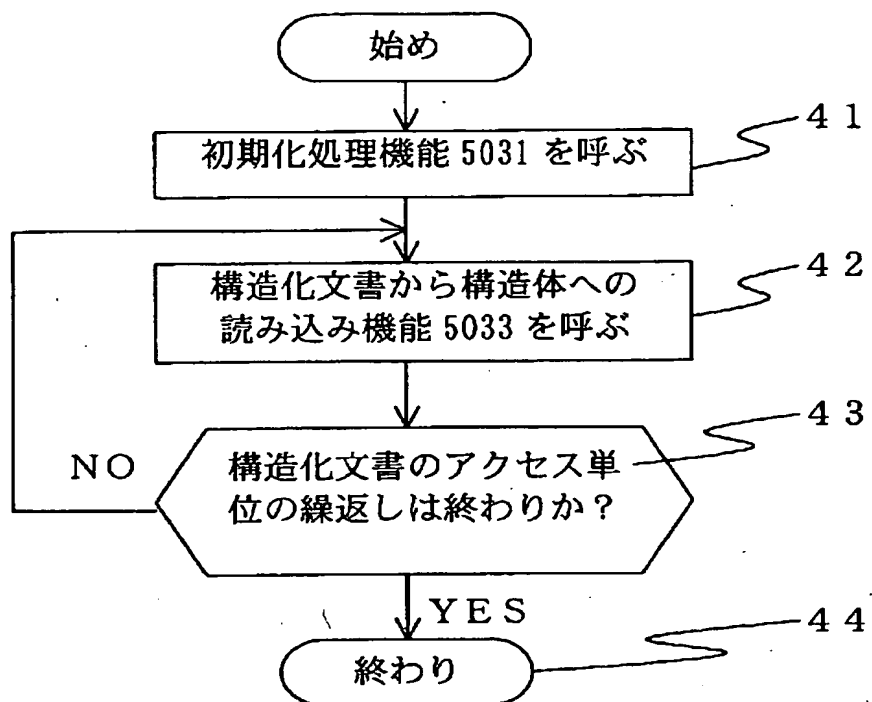
【図 1 1】

図 1 1

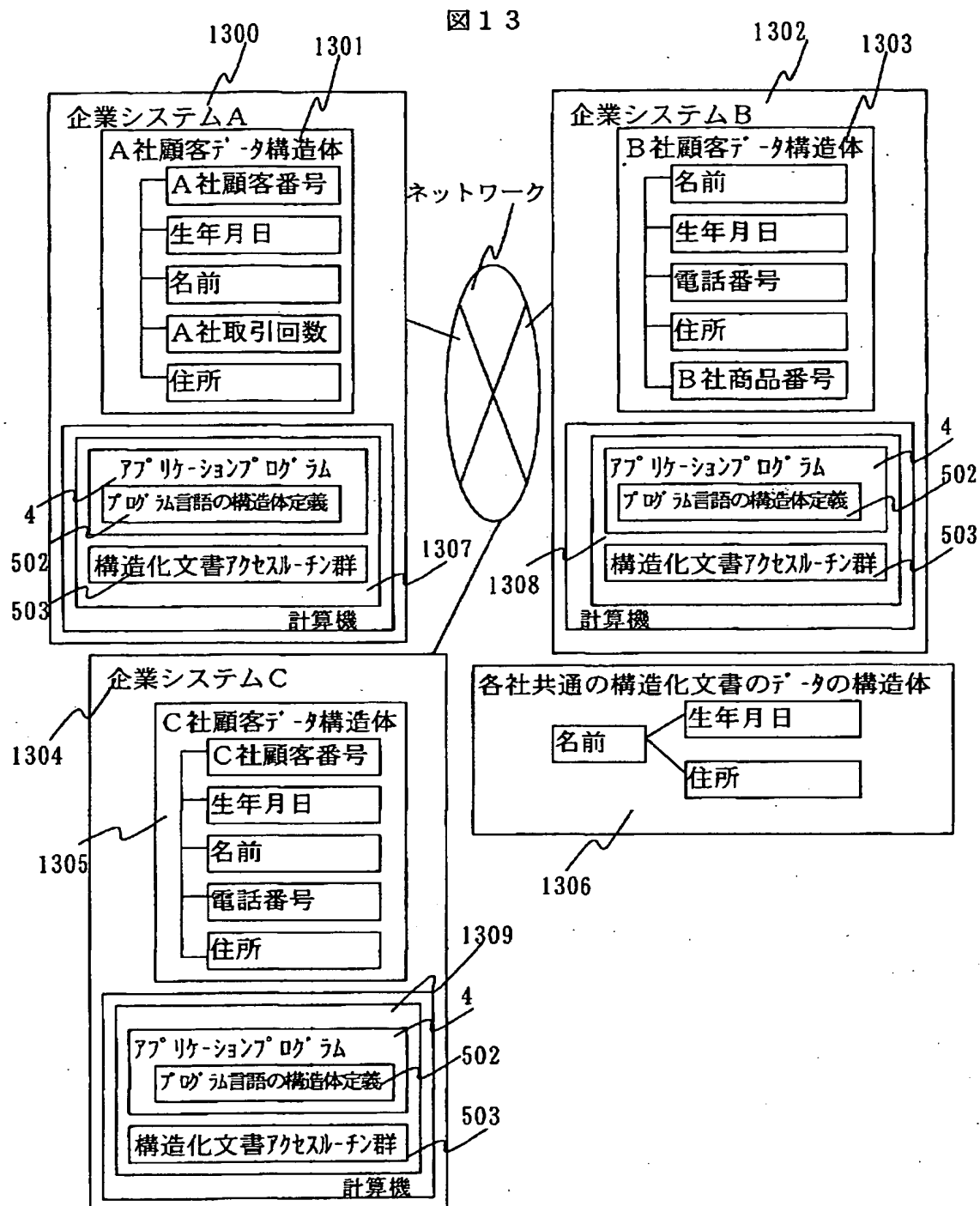


【図12】

図12

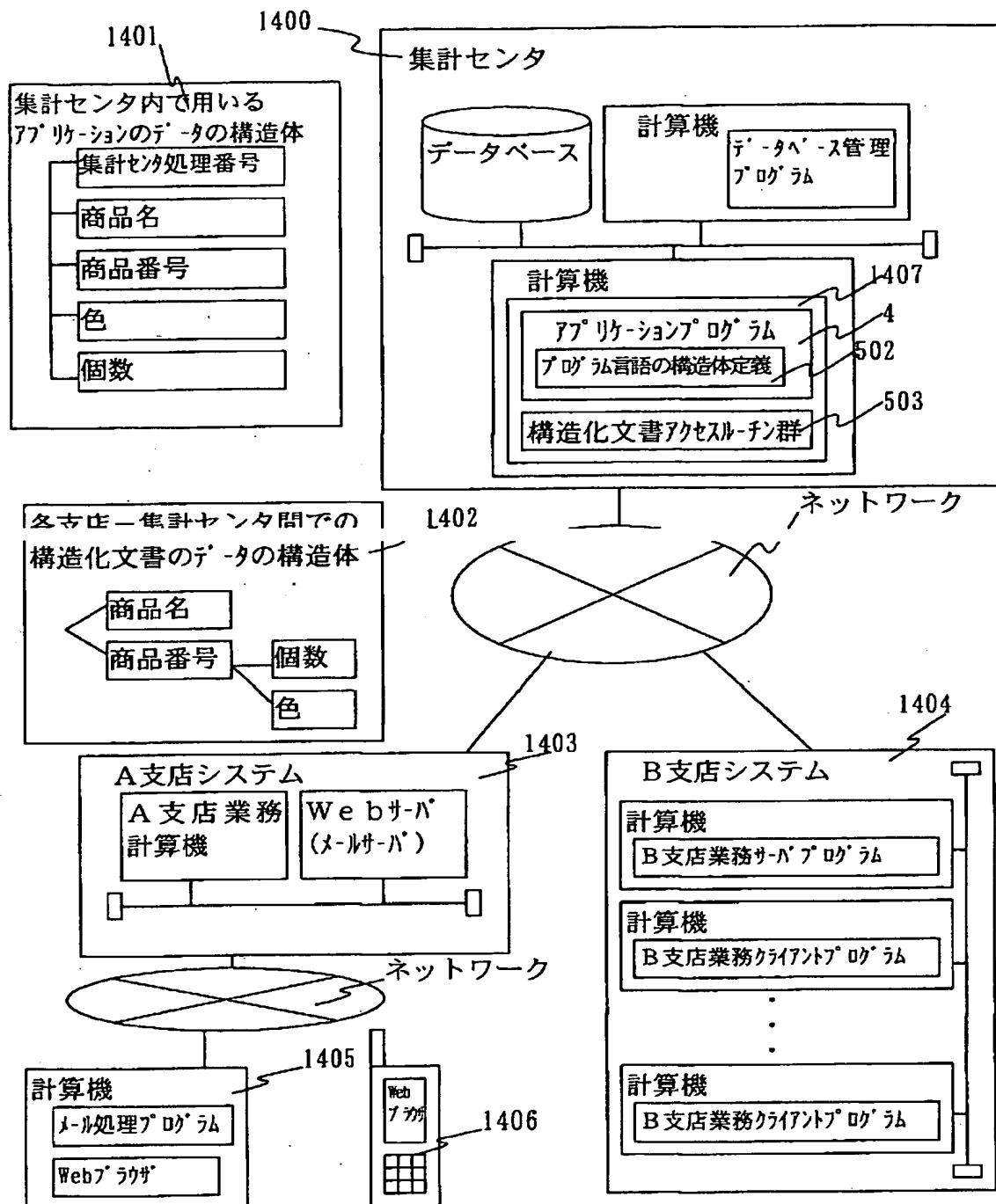


【図 13】



【図 14】

図 14



【書類名】 要約書

【要約】

【課題】

構造化文書のデータを扱うアプリケーションを作成する場合、従来は、動的に木構造に展開したり、イベントによって呼び出されるコールバック・ルーチンを駆使したりするプログラミングをする必要があったが、処理が複雑であるだけでなく、COBOLのようなポインタの概念の無いプログラム言語では実現が困難であるという問題があった。

【解決手段】

データ転記処理部5を用意し、これに処理対象の構造化文書の文書構造定義情報1、プログラム言語の構造体定義情報2、及び、文書構造定義と構造体定義との対応情報3を与えておく。アプリケーションプログラム4からデータ転記処理部5に読み書きの要求を出すだけで、構造化文書6とプログラム言語の構造体との間で要素毎のデータ転記が実現する。

【選択図】 図1

出 願 人 履 歴 情 報

識別番号 [000005108]

1. 変更年月日 1990年 8月31日

[変更理由] 新規登録

住 所 東京都千代田区神田駿河台4丁目6番地  
氏 名 株式会社日立製作所